Cálculo de Fluencia en LAGO

I. Instalación de CORSIKA

H. Asorey[*1]

[1]Centro Atómico Bariloche (CNEA)

2013

Resumen

En esta nota técnica, la primera de una serie destinada a calcular la fluencia de fotones producida por un GRB y el fondo de radiación producido por los rayos cósmicos, se describen los pasos necesarios para instalar y configurar el código de simulación CORSIKA con las opciones necesarias para nuestro estudio.

1. Introducción

CORSIKA [1] (COsmic Rays SImulations for KAscade) es un código diseñado para realizar simulaciones detalladas de las lluvias atmosféricas extendidas (en adelante, EAS, por sus siglas en ingles *Extensive Air Showers*). El código permite simulas las cascadas producidos por numerosos tipos de primarios(protones, núcleos, electrones, fotones, etc.), y ha sido ampliamente utilizado para validar las observaciones en numerosos experimentos y observatorios en un amplio rango de energías. A las energías donde fuera posible, los valores utilizados para las secciones eficaces de interacción y vidas medias han sido obtenidos y actualizados con los resultados obtenidos en los aceleradores de partículas.

La amplia aceptación del código y sus resultados, así como su continuo desarrollo y mantenimiento, hacen que esta sea la opción más confiable a la hora de realizar las simulaciones necesarias para LAGO. Utilizaremos para ello la última versión disponible, CORSIKA v7.3500 [2], lanzada en Diciembre de 2012.

[*] asoreyh@cab.cnea.gov.ar

2. Obtención del código

La obtención del código puede hacerse a partir de la página de descarga del mismo, Access to CORSIKA. Esto requiere del registro de la IP de la máquina desde la cual se realizará la descarga, para lo cual es necesario previamente enviar un mail a T. Pierog. Sin embargo, con el objeto de simplicar la descarga a los usuarios de LAGO, se ha dispuesto una copia del cógido fuente y las librerías del mismo en una carpeta compartida en Dropbox. Para descargar el mismo, debe seguirse el siguiente enlace: CORSIKA v73500 installer.

3. Instalación de CORSIKA

En lo sucesivo se supone que el código será instalado y ejecutado en un sistema, máquina de escritorio o cluster, basado en Unix o Linux. En particular, se dan los nombres de los paquetes que instalan las librerías necesarias para la compilación de CORSIKA para un sistema Debian (p. ej. Ubuntu).

3.1. Requisitos del sistema

En general, las librerías y aplicaciones requeridas por CORSIKA son pocas, y usualmente ya se encuentran instaladas en la mayoría de los sistemas de cálculo. Para instalar los paquetes requeridos, desde una terminal ejecutar:

```
1 $ sudo apt-get install build-essential gfortran-4.4 gcc-4.4
```

Adicionalmente, si se requiere el uso de las herramientas para dibujar la evolución de las lluvias, se deberán instalar además los siguientes paquetes:[1]

```
1 $ sudo apt-get install libgd2-xpm libgd2-xpm-dev
```

Una vez completados estos dos pasos, es necesario ahora forzar al sistema para utilizar las versiones 4.4 de los compiladores asociados a los compiladores `gcc`, ya que se han observado algunos problemas de incompatibilidad con compiladores de la serie 4.6. Hay varias formas de lograr esto, una de ellas es simplemente reemplazar los enlaces simbólicos a los compiladores de la siguiente forma:

```
1 $ sudo ln -s /usr/bin/gfortran-4.4 /usr/bin/gfortran
2 $ sudo rm /usr/bin/gcc
3 $ sudo ln -s /usr/bin/gcc-4.4 /usr/bin/gcc
4 $ sudo rm /usr/bin/g++
5 $ sudo ln -s /usr/bin/g++-4.4 /usr/bin/g++
```

y luego, al finalizar la compilación de CORSIKA, volver a las versiones originales:

[1] La instalación de estos paquetes podría requerir la instalación automática de varios otros paquetes adicionales

```
$ sudo rm /usr/bin/gfortran
$ sudo ln -s /usr/bin/gfortran-4.6 /usr/bin/gfortran
$ sudo rm /usr/bin/gcc
$ sudo ln -s /usr/bin/gcc-4.6 /usr/bin/gcc
$ sudo rm /usr/bin/g++
$ sudo ln -s /usr/bin/g++-4.6 /usr/bin/g++
```

3.2. Instalación

Esta nota debe ser considerada como una guía básica para la instalación de CORSIKA y la extracción de las opciones necesarias para el cálculo de fluencias. Hacer una descripción completa de cada una de las opciones y su significado exceden por completo el alcance de la presente. La guía completa de instalación puede verse en el archivo `INSTALL`, localizado en el directorio de instalación de CORSIKA, y en la guía del usuario del programa [3].

En primer lugar, es necesario descomprimir el paquete de instalación:

```
$ tar xfv corsika-73500.tar.gz
```

lo cual creará el directorio `corsika-73500`. Tambien es necesario descomprimir las rutinas que permiten utilizar otros modelos atmosféricos adicionales a los incluídos en CORSIKA. Estas librerías, programadas y mantenidas por K. Bernhlor, se encuentran comprimidas en el directorio `bernhlor` y deben ser extraídas antes de proceder a la instalación:

```
$ cd corsika-73500
$ cd bernlohr
$ tar xfv bernlohr-1.46.tar.gz
$ cd ..
```

En general, el proceso habitual de instalación desde las fuentes en un sistema Linux se hace siguiendo la habitual secuencia `configure-make-make install`. Sin embargo, CORSIKA utiliza un script que automatiza la extracción de los distintos componentes del código y posteriormente procede a la compilación del mismo. Para ello, entramos al directorio de instalación y ejecutamos el script llamado `coconut`:

```
$ ./coconut
```

El script realizará una serie de preguntas que nos guiarán a lo largo del proceso de configuración, compilación e instalación. Todas las preguntas poseen varias opciones posibles, y una de ellas es la opción por defecto, usualmente la más común, aunque no necesariamente la que usaremos para compilar al código de acuerdo a nuestras necesidades. En la figura 1 se muestra una captura de pantalla del inicio del proceso de configuración.

A continuación, se describe el primer conjunto de preguntas que deberán contestarse para configurar CORSIKA:

1. **Arquitectura del sistema**: Si el sistema de es de 64 bits[2] `coconut` preguntará si debe compilar CORSIKA en la forma nativa del sistema (64 bits), o forzar el modo de compatibilidad a 32 bits. En general, la mayoría de los clusters modernos funcionan con una arquitectura de 64 bits, y por lo tanto podemos utilizar la versión de 64 bits para las librerías.
2. **Modelo hadrónico a altas energías**: A las más altas energías de primario, $E_p \gtrsim 10^{16}$ eV, los resultados de las simulaciones dependerán del modelo hadrónico utilizado para las mismas. Este es un fenómeno conocido y ha sido ampliamente debatido en la literatura (ver por ejemplo [4]). Sin embargo, a energías del primario más bajas, donde las simulaciones pueden cotejarse con los experimentos realizados en aceleradores, la elección del modelo hadrónico utilizado no tiene un impacto significativo en la distribución de secundarios al nivel del detector [5]. Al momento de escribir este informe técnico, son dos los modelos hadrónicos que incluyen las primeras mediciones realizadas en el LHC: los llamados QJSJET-II-04 [6] y EPOS-LHC (v3400) [7]. Para este análisis, usaremos el modelo QJSJET-II-04, seleccionando por ende la opción 5.
3. **Modelo hadrónico a bajas energías**: Esta librería es la responsable del manejo de las interacciones a bajas energías y los decaimientos de las partículas inestables. En este caso, elegiremos la opción por defecto (GHEISHA-2002 [8]).
4. **Geometría del detector**: La geometría del detector influye directamente en la forma de simular el flujo angular de primarios. Hay tres opciones posibles: arreglo plano de detectores (el flujo de primarios es $I \propto \sin\theta\cos\theta$); detector volumétrico (el flujo de primarios es $I \propto \sin\theta$); y cadena vertical de detectores, como los utilizados en IceCube, donde el flujo depende de las características de la cadena. Para LAGO, utilizaremos la opción 2 (detector volumétrico). Si en cambio hubiera varios detectores WCD distribuidos en un mismo sitio, utilizaremos la opción 1, ya que en general la separación entre detectores es mucho mayor que la altura del detector.

Llegado a este punto, las opciones de precompilación han sido seleccionadas [3]:

```
options:   QGSJETII VOLUMEDET GHEISHA TIMEAUTO.
```

Es necesario ahora seleccionar las opciones adicionales. Seleccionaremos sólo las siguientes opciones:

4 - external atmosphere functions (table interpolation)

7 - shower PLOT version (PLOTSH) (only for single events)

a - CURVED atmosphere version

j - COMPACT particle output file

l - LPM-effect without thinning

[2] Para verificar la arquitectura del sistema puede usarse el comando `uname -a`.
[3] `VOLUMEDET` indica que se ha seleccionado una geometría de detector volumétrica

[~/corsika-73500]$ ./coconut
Welcome to COCONUT (v3.1)
-- the CORSIKA CONfiguration UTility --
create an executable of a specific CORSIKA version
Please read the documentation for a detailed description
of the options and how to use it.
Try './coconut -h' to get some help about COCONUT
Use './coconut --expert' to enable additional configuration steps.
(press 'Enter' to select an option followed by "[DEFAULT]" or "[CACHED]")
Which high energy hadronic interaction model do you want to use ?
1 - DPMJET 2.55
2 - EPOS LHC
3 - NEXUS 3.97
4 - QGSJET 01C (enlarged commons) [DEFAULT]
5 - QGSJETII-04
6 - SIBYLL 2.1
7 - VENUS 4.12
r - restart (reset all options)
x - exit make
(only one choice possible):

Figura 1: Salida del script de instalación `coconut`, donde puede verse el mensaje de bienvenida y la primer pregunta para un sistema de 32 bits.

v - EFIELD version for electrical field in atmosphere

Llegado a este punto, las opciones seleccionadas son las siguientes:

```
options:   QGSJETII VOLUMEDET GHEISHA TIMEAUTO
selection: EFIELD COMPACT CURVED ATMEXT PLOTSH LPM BERNLOHRDIR
```

Seguiremos adelante presionando la tecla enter y confirmando la selección de las opciones:

```
Are you sure you want to continue with these current option selection:
    EFIELD COMPACT CURVED ATMEXT PLOTSH LPM BERNLOHRDIR
    yes or no ? (default: yes) > yes
```

Finalmente, confirmamos el inicio de la compilación utilizando la opción por defecto (presionando enter una vez más), para comenzar el proceso de extracción y compilación del código. Luego de unos instantes, la compilación finaliza confirmando la creación de los ejecutables en el directorio `corsika-73500/run`:

```
[...]
--> "corsika73500Linux_QGSII_gheisha" successfully installed in :
    /home/asoreyh/corsika-73500/run/

--> You can run CORSIKA in /home/asoreyh/corsika-73500/run/ using
for instance :
    ./corsika73500Linux_QGSII_gheisha < all-inputs > output.txt
[...]
```

En el directorio `run` se encuentran los ejecutables y las librerías necesarias para el cálculo de secciones eficaces.

3.3. Instalación en otros nodos

Todos los archivos necesarios para ejecutar CORSIKA se encuentran en el directorio `run`. Por lo tanto, en clusters o sistemas formados por múltiples máquinas con la misma arquitectura, no es necesario recompilar el código: sólo es necesario copiar el contenido del directorio de ejecución al resto de los nodos de la red, si lo que se desea es ejecutar localmente la simulación.

4. Ejecución de prueba

Una vez finalizada la instalación se sugiere la ejecución de una de las simulaciones de prueba provistas junto con el código de simulación. La misma se encuentra en el directorio `run` en el archivo `all-inputs`. El ejecutable lee el archivo de configuración a partir del canal STDIN:

```
1 $ cd run
2 $ ./corsika73500Linux_QGSII_gheisha < all-inputs > output.lst
```

Al finalizar la simulación, se habrá creado en el mismo directorio un archivo binario llamado `DAT000001` que contiene los resultados de la simulación. Próximamente describiremos las herramientas disponibles para el análisis de dicho archivo. El contenido del archivo `output.lst`, contiene salidas que produce el programa durante la simulación, y da una idea del desarrollo de la misma, así como características generales del avance de la corrida. Este archivo contiene información relevante para controlar el estado actual de la corrida que se este ejecutando.

5. Conclusiones

En esta breve nota técnica se han descripto los pasos necesarios para la correcta instalación del código de simulación CORSIKA v7.3500 y las librerías auxiliares requeridas. También se detallaron las opciones relevantes para realizar simulaciones en LAGO, junto con un ejemplo de funcionamiento.

6. Agradecimientos

El autor desea expresar su agradecimiento a I. Sidelnik por la cuidadosa lectura del manuscrito y sus sugerencias para mejorar el mismo.

Referencias

[1] D. Heck, J. Knapp, J. Capdevielle, G. Schatz, and T. Thouw, "CORSIKA : A Monte Carlo Code to Simulate Extensive Air Showers," *FZKA*, vol. 6019, pp. 1–98, 1998.

[2] D. Heck and T. Pierog, *Extensive Air Shower Simulation with CORSIKA: A User's Guide*. 7.3500 ed., 2012.

[3] D. Heck and T. Pierog, *Extensive Air Shower Simulation with CORSIKA: A User's Guide*. Karlsruhe, Germany: Karlsruhe Institute of Technology - KIT, 6.990 ed., 2012.

[4] D. D'Enterria, R. Engel, T. Pierog, S. Ostapchenko, and K. Werner, "Constraints from the first LHC data on hadronic event generators for ultra-high energy cosmic-ray physics," *Astroparticle Physics*, vol. 35, pp. 98–113, Sept. 2011.

[5] S. Ostapchenko and H. Asorey, "High-energy hadronic model dependence at low energy of primaries," 2011.

[6] S. Ostapchenko, "Monte Carlo treatment of hadronic interactions in enhanced Pomeron scheme: QGSJET-II model," *Physical Review D*, vol. 83, p. 014018, Jan. 2011.

[7] T. Pierog and K. Werner, "To be published." 2013.

[8] H. C. Fesefeldt, "GHEISHA program, Technical Report PITHA 85-02," tech. rep., III Physikalisches Institut, Aachen, Germany, 1985.